Année 2023-2024 — 1ère session – 18 Janvier 2024

BORDEAUX INP Enseirb-Matmeca

PG109 – Programmation Impérative

1ère année Télécom

Joachim Bruneau-Queyreix
Kylian Desier
Guillaume Mercier
Augusta Mukam
Philippe Swartvagher

Durée de l'examen : 1 h

Documents autorisés	☐	sans document	☒
Calculatrice autorisée	☐	non autorisée	☒

Nom et Prénom: _____

- **N'oubliez pas d'écrire votre nom ci-dessus !**
- **Il est impératif de répondre dans les espaces prévus à cet effet : tout ce qui dépasse ne sera pas pris en compte lors de la correction.**
- **L'orthographe, la grammaire, la conjugaison et la syntaxe seront pris en compte lors de la correction.**
- **Écrivez le code comme si vous écriviez dans un fichier source : la syntaxe, le style (indentation, noms de variables cohérents, ...) seront pris en compte lors de la correction. Il va sans dire que le code que vous écrivez doit compiler.**
- **N'oubliez pas d'écrire votre nom ci-dessus !**

Exercice 1 – QCM

Chaque question possède une et une seule bonne réponse.

Barème :

- 1 point par bonne réponse
- **-0,5 point par mauvaise réponse**
- 0 point si absence de réponse

On considérera que chaque extrait de code présenté est correctement intégré dans une fonction `main()` et que tous les `#include` nécessaires sont présents.

1. Qu'affiche le code suivant ?

```c
int a;
printf("%d\n", a);
```

   - ☐ Rien, parce que la compilation échoue
   - ☐ Rien, parce que l'exécution échoue
   - ☐ Rien.
   - ☐ `0`
   - ☐ `1`
   - ☐ Il n'est pas possible de prédire la valeur qui sera affichée

2. Comment peut-on connaître le nombre d'éléments dans n'importe quel tableau ?
   - ☐ Avec la fonction `strlen()`
   - ☐ On ne peut pas
   - ☐ Avec l'opérateur `sizeof`
   - ☐ On parcourt le tableau jusqu'à atteindre une valeur particulière qui indique la fin du tableau
   - ☐ Avec la fonction `len()`

3. Qu'affiche le code suivant ?

```c
int t[4] = {100, 200, 300, 400};
int* u = t;
int* v = u + 2;
int* z = &u[1];
v[1] = 500;
printf("%d\n", z[1]);
```

   - ☐ `200`
   - ☐ `300`
   - ☐ `400`
   - ☐ `500`
   - ☐ Une autre valeur

4. Quelle ligne du code suivant pose problème ?

```c
1 char t[] = "Bonjour\n" ;
2 char* s = "Salut!";
3 t[1] = 'Z';
4 s[1] = 'Z';
5 s = t;
6 s[1] = 'A';
```

- ☐ Aucune
- ☐ Ligne 3
- ☐ Ligne 4
- ☐ Ligne 5
- ☐ Ligne 6

5. Qu'affiche le code suivant ?

```
float x = 'b';
printf("%f\n", x);
```

- ☐ `b`
- ☐ Rien, une erreur se produit à la compilation ou à l'exécution
- ☐ `b.000000`
- ☐ `98.000000`

Exercice 2 – Questions de cours

1. Quelle condition faut-il utiliser dans le `if` pour tester l'égalité des variables `a` et `b` ?

```
float a, b;

// ... des opérations sur a et b ...

if (???)
{
  printf("a et b ont la même valeur : %f = %f\n", a, b);
}
```

.....

2. Soient `a` et `b` deux variables de type `int`.
   (a) Quelle est la différence entre `if (a & b)` et `if (a && b)` ?

   .....

   .....

   (b) Donner un exemple de valeurs pour `a` et `b` qui illustre cette différence.

   .....

   .....

3. Avec le code suivant, pourquoi peut-on définir le type `struct foo_s`, mais pas `struct bar_s` ?

```
struct foo_s { // Structure valide
    struct foo_s* s;
    // ...
};

struct bar_s { // Structure invalide
    struct bar_s s;
    // ...
};
```

.....

.....

.....

4. (a) Pourquoi le code suivant générera une erreur de segmentation ?

```
char* s1 = "Carottes râpées";
char* s2;
strcpy(s2, s1);
```

.....

.....

(b) Que faudrait-il faire pour corriger cette erreur ?

.....

.....

.....

Exercice 3 – Écrivons un peu de code

On souhaite écrire un programme qui affiche le maximum des nombres entiers qui lui sont passés en paramètres. Par exemple :

```
./prog 3 7 8 2 1
8
```

1. Écrire une fonction `max_tab()` qui prend en paramètre un tableau de chaînes de caractères représentant des nombres (et peut-être d'autre(s) paramètre(s)) et renvoie le plus grand nombre dans le tableau, en utilisant le type `int`. On considérera que le tableau possède toujours au moins un élément et que les chaînes de caractères du tableau représentent toujours des nombres entiers.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

```
14
15
16
17
18
19
20
```

2. Écrire le reste du code nécessaire pour avoir un tel programme : instructions de préprocesseur, fonction `main()`, *etc.* Seule la fonction `max_tab()` sera omise, son emplacement dans le code source sera signifié par un commentaire. On considérera que le programme est toujours bien utilisé (nombre de paramètres, ...), ce n'est donc pas la peine de vérifier le nombre de paramètres du programme.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

Exercice 4 – Une interface pour un jeu de pendu

Dans cet exercice, il est question d'implémenter une interface pour gérer un jeu de pendu. Voici un exemple de partie :

```
./pendu telecom

# le contenu du terminal est effacé (pour ne pas voir le mot à trouver !)

Il faut trouver: -------
Proposez une lettre : a
Il n'y a pas de lettre 'a' dans le mot à trouver...
Il faut trouver: -------
Proposez une lettre : e
Oui ! La lettre 'e' est présente 2 fois !
Il faut trouver: -e-e---
Proposez une lettre : c
Oui ! La lettre 'c' est présente 1 fois !
Il faut trouver: -e-ec--
Proposez une lettre : o
Oui ! La lettre 'o' est présente 1 fois !
Il faut trouver: -e-eco-
Proposez une lettre : e
La lettre 'e' a déjà été trouvée...
Il faut trouver: -e-eco-
Proposez une lettre : m
Oui ! La lettre 'm' est présente 1 fois !
Il faut trouver: -e-ecom
Proposez une lettre : t
Oui ! La lettre 't' est présente 1 fois !
Il faut trouver: te-ecom
Proposez une lettre : l
Oui ! La lettre 'l' est présente 1 fois !
Bravo ! Vous avez trouvé le mot en 6 coups ! Le mot était 'telecom'.
```

L'interface utilisera cette structure pour gérer l'état du jeu :

```
struct pendu_s {
    char* mot; // mot à trouver
    int len_mot; // nombre de lettres dans le mot à trouver
    int* lettres_trouvees; // tableau de taille len_mot qui indique quelles
       ↪ lettres ont déjà été trouvées
    int nb_essais; // nombre de propositions de lettres
    int nb_lettres_trouvees; // nombre de lettres déjà trouvées dans le mot
};
```

Le tableau `lettres_trouvees` contient des booléens : si `lettres_trouvees[i]` vaut `vrai`, cela signifie que la lettre `mot[i]` a été trouvée.

1. Écrire la fonction qui permet d'initialiser la structure `struct pendu_s`. On considérera que les allocations de mémoire dynamiques réussissent toujours : ce n'est donc pas nécessaire de gérer les cas où une erreur surviendrait.

```
1 struct pendu_s* init_pendu(char* mot)
2 {
3
4
5
6
7
```

```
8
9
10
11
12
13
14
15 }
```

2. Écrire la fonction qui permet de libérer la mémoire allouée par la fonction `init_pendu()`.

```
1 void pendu_libere(struct pendu_s* pendu)
2 {
3
4
5 }
```

3. Écrire la fonction qui permet de proposer une lettre. Si la lettre proposée appartient au mot à trouver, le tableau `lettres_trouvees` est mis à jour en conséquence. Cette fonction renvoie le nombre de lettres découvertes ou `-1` si la lettre a déjà été trouvée. On ne se préoccupera pas des différences de casse (lettre majuscule ou minuscule). Ne pas oublier de mettre également à jour les différents compteurs.

```
1 int pendu_tester_lettre(struct pendu_s* pendu, char lettre)
2 {
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24 }
```

4. Écrire la fonction qui permet de savoir si on a gagné, *i.e.*, si toutes les lettres du mot ont été découvertes.

```
int pendu_gagne(struct pendu_s* pendu)
{



}
```

5. Écrire la fonction qui affiche le mot avec les lettres découvertes et en remplaçant les autres lettres par un tiret -.

```
void pendu_affiche_mot(struct pendu_s* pendu)
{









}
```

6. Écrire la fonction qui affiche le message de victoire en indiquant le nombre de coups utilisés.

```
void pendu_affiche_victoire(struct pendu_s* pendu)
{



}
```

7. (Bonus) En utilisant les fonctions précédentes, compléter la fonction `main()` qui permet de jouer au pendu. Le mot à trouver est passé comme paramètre du programme. On considérera que le programme est toujours bien utilisé (nombre de paramètres, ...) et que l'utilisateur saisit toujours une lettre.

   Pour effacer le contenu du terminal, on pourra utiliser le code suivant :

```
printf("\e[1;1H\e[2J");
```

   Pour demander à l'utilisateur de saisir une lettre, on pourra utiliser le code suivant :

```
printf("Proposez une lettre : ");
char lettre;
scanf(" %1c", &lettre);
```

```
int main(
{
}
```